(1)

SUBMICRON SYSTEMS ARCHITECTURE

SEMIANNUAL TECHNICAL REPORT

Sponsored by
Defense Advanced Research Projects Agency
ARPA Order Number 3771

Monitored by the
Office of Naval Research
Contract Number N00014-79-C-0597

5078:TR:83

Computer Science
California Institute of Technology

April 1983

SUBMICRON SYSTEMS ARCHITECTURE

Semiannual Technical Report

California Institute of Technology

5078:TR:83

April 1983

Sponsored by
Defense Advanced Research Projects Agency
ARPA Order Number 3771

Monitored by the
Office of Naval Research
Contract Number N00014-79-C-0597

SUBMICRON SYSTEMS ARCHITECTURE

Computer Science
California Institute of Technology

1. OVERVIEW AND SUMMARY

1.1 Scope of this Report

This document reports the research activities and results for the period 16 October 1982 - 15 April 1982 under the Defense Advanced Research Project Agency (ARPA) Submicron Systems Architecture Project.

1.2 Objectives

The central theme of this research is the architecture and design of VLSI systems appropriate to a microcircuit technology scaled to submicron feature sizes, and includes related efforts in concurrent computation and VLSI design. The scope of the research is discussed in greater detail in our previous technical report [5052:TR:82].

1.3 Highlights

The highlights of the previous 6 months are:

(1) The design of a tree machine processor, called "mosaic," was completed and verified in November and December 82. This 16-bit processor is only 1440 by 1436 lambda, or 2 million square lambda (MSL) in size, including 4 bidirectional communication channels. The design was sent to MOSIS for fabrication on 6 January 83, and returned from fabrication 34 days later. The chips returned were tested immediately, and execute code correctly. The maximum clock rate, which is also the storage cycle time, is 7. 0 MHz, very good performance for the technology used, and achieved in part through using a "hot clock" discipline with clock-and bootstrap drivers. Continued testing revealed a very simply fixed microcode bug, and the design appears to be almost ready for production in larger quantities.

(2) A new and simplified software system for the cosmic cube homogeneous machine has been developed. The intermediate host has been redesigned, the 6-cube construction is virtually complete, and this 64 processor machine is being tested. Routine operation of the full system is expected within the next month.

(3) A fifo buffered one-to-one communication chip for future experimental homogeneous machines was tested in March 83 and found to work correctly at a 6 MHz clock rate. The first iteration of this design tested in December 82 was only partially functional due to a ratio error.

(4) A new proof rule for procedures has been discovered that is much more general and easy to use than existing ones.

A self-timed N-servers mutual exclusion circuit has been design using a systematic method to derive circuits from programs.

2. ARCHITECTURAL EXPERIMENTS

2.1 Overview

We have four architectural experiments in various phases of design and construction. They are all ensembles of identical, concurrently operating, and regularly interconnected elements that communicate by message passing. Our priority in these efforts has been to apply VLSI technology to achieve substantial advances in cost/performance in a limited set of computationally demanding tasks. Generally the prototypes will exhibit a performance that for a limited set of problems scales linearly with the number of elements, or cost, and so are benchmarked in terms of cost/performance. These experimental machines differ in the size (complexity) of the repeated element, interconnection plan, MIMD vs SIMD, and their application span. The complexity of the elements is most readily expressed in million square lambda units.

These experiments, the machine organizations, their current status, and application span, can be summarized as follows:

(1) Cosmic-cube is a homogeneous machine with elements interconnected in a Boolean n-cube. Cosmic elements are of medium size for this class of machine, about 140 MSL, and consist of an Intel 8086 processor with 8087 floating point coprocessor, 128K bytes of primary storage, and 6 bidirectional self-timed communication channels. At 140 MSL, and 78 "off the shelf" chips, the nodes of this machine are something of a "hardware simulation" of the kind of node that could be made as a single chip with 1 micron MOS technology, but even this prototype produces very good performance and about a factor of 10 advance in cost/performance over most mainframes. A 2-cube prototype has been running concurrent programs since July 1982, and has been used for program development. Construction of a 6-cube machine is nearly complete, and the machine is being tested in stages. This machine appears to be very cost effective for a variety of regular matrix, grid point, and n-body problems, as well as for less regular computations such as SPICE simulation.

(2) Mosaic-mesh is a homogeneous machine with elements interconnected in a two-dimensional mesh. Mosaic elements are small for this class of machine, and consist of a mosaic processor (2 MSL) with as much primary storage as a single chip permits. For example, a 6 mm square chip in 3 micron nMOS technology, 16 MSL, will accommodate a mosaic processor, bootstrap ROM, and 4K bytes of dynamic RAM. The mosaic processor element is designed and tested, RAM test chips have been fabricated, and a full-size RAM element is being designed. This project involves a number of supporting efforts in testing to allow production of these chips in quantities of several thousands. This system is capable of and offers a factor of about 5 in cost/performance over some of the most regular computations that can be performed on cosmic cube, and that do not require large amounts of storage per node.

(3) Mosaic-tree could be regarded as a homogeneous machine with very small, indeed, marginally small elements, interconnected as a binary tree. It also employs the mosaic processor (2 MSL) with a small amount of primary storage. An area equal to that of the processor devoted to storage allows about 0.5K

bytes of primary storage, and the total node complexity of 4 MSL allows 4 nodes per chip. A prototype mosaic tree with mosaic processors connected to 8K bytes of off-chip RAM will be operating later this spring, shortly after the next batch of mosaic chips arrives from MOSIS. The logistics of possibly building a useful size prototype of this system are closely tied to mosaic-mesh. As previously reported, even with this small amount of storage and the tree interconnection, a 1000 chip mosiac-tree would be capable of performing an interesting variety of numerical and graph computations with very high performance.

(4) Super-mesh is an SIMD machine in the early stages of design. Its elements are expected to be quite small, about 1 MSL, containing only registers, serial floating point arithmetic, and neighbor communication. An instruction cycle of this machine requires several more clock cycles than its mantissa word length, and a microcode control word is transmitted serially for each instruction cycle. The physical design of this machine employs a deliberate skew in the internode communication and instruction broadcast to allow it to be extended to any size, but its interconnection is limited to a mesh. This machine might be regarded as a very efficient implementation of a computational or systolic array, with the advantage that the system is programmable for a variety of applications. A study of applications has revealed a number of limitations in this model that suggests that a useful machine will have to have either a large amount of storage per node or a very high bandwidth access mechanism to a secondary storage system.

2.2 Cosmic Cube

2.2.1 Hardware

William C Athas, Pete Hunter, Chuck Seitz

Design and prototyping efforts for this project were reported in our previous semiannual technical report. Hardware efforts for the Cosmic Cube have centered in the past 6 months on logistics for assembling and testing the 6-cube (64 processor) machine.

PC boards for the processors were delivered in February, and almost all of the integrated circuit and packaging parts have been delivered in the January to March period. The processor boards have been populated with sockets and components at the rate of two boards per day, and are now almost all assembled. These boards are being tested in a test fixture consisting of a cosmic cube Intermediate Host (IH) and a degenerate cube.

The intermediate host has been redesigned and simplified to a standard 86/12A multibus system with a single wire-wrapped Cube Life Support (CLS) board. CLS is the umbilical cord for Cosmic Cube, providing clocks, RAM refresh interrupts and FIFO message ports from CLS to the Cosmic Cube Corner Processor (CCCP).

Wire-wrapping of the backplane for the Cosmic Cube was started 2 weeks ago and is about half completed.

2.2.2 Software

William C Athas, Michael O Newton, Chuck Seitz

All the low-level, or machine intrinsic level, software for the cosmic cube has been redesigned and simplified, while also increasing its generality. These improvements were motivated by the impromtu character of the first try at this software, and upon two assumptions. First of all, we expect the cosmic cube to be followed by machines in the next several years that have different numbers and kinds of elements, and want to make as much of our software portable to new machines as possible. Secondly, it it clear that the cosmic cube will be able to support many different programming environments. The earlier software tried to put too much operating system in read-only storage, only to encourage users to bypass it. So, the machine intrinsic (R/O storage) part is now stripped to the bare essentials -- initialization and boot loading.

An initialization protocol has been defined for making the cosmic cube presentable to the user after a hardware reset. Initialization starts execution of code in firmware to initialize message ports and poll for incoming messages. An initialization packet from the IH is diffused through the cosmic cube starting at the corner processor. This packet contains information regarding the logical size of the cube, size of the FIFO queues, software version, and size of subsequent packets. After this initialization each processor knows where it resides in the cube with respect to the corner processor and now executes a small boot loader also in firmware. The boot loader ordinarily loads into RAM a more elaborate loader provided by the user. At this point further configuration is done only by the user at his or her discretion and risk. We believe this scheme to be completely flexible, and it is compatible with the retargetable C Compiler and Unix derivative systems developed by High Energy Physics.

Work has also started on simplifying user interfaces to the cosmic cube. We plan to retain the Intermediate Host (IH) based on the Intel 86/12A multibus system, expanding it as necessary to provide interfaces to networks and to other computers. We are in the process of replacing the initial large firmware monitor of the IH with a small intialization and bootstrap loader that provides only rudimentary hardware debug facilities. The user can define his or her own run time interface to the cosmic cube corner processor, if desired. The new monitor for the IH is written in C, and it is intended to be easy to port useful code such as network protocols to the IH.

Meanwhile, work is continuing on developing a simple multiprogramming operating system to run in the cube. We have also recently been doing some experiments in using IBM Personal Computers for program development, and possibly as an IH, for the cosmic cube. The IBM PCs also use 8086/8087s, and include a Pascal compiler that produces easily linked 8086 assembly code. Upon first analysis, these IBM PCs appear to be an attractive approach to allowing system and application programming for the cosmic cube to be done in Pascal.

2.2.3 Designs for a Boolean 10-cube Homogeneous Machine

Several preliminary designs for a Boolean 10-cube machine have been developed, stimulated by a commitment from Intel to contribute chips of their manufacture and strong possibilities of funding, some already committed, for the construction and programming of such a system. This 1024 processor system would be a direct evolution of cosmic cube, and source program compatible with it.

Storage per node is to be increased from 128K bytes to 256K bytes, expandable to 512K. Such a large amount of primary storage benefits greatly from error correction rather than the simple parity detection used in cosmic cube. Soft errors can be expected to occur at a rate of one per several minutes in a system with 256M bytes of dynamic RAM, considerably less than the running time of most programs. Error correction also greatly simplifies the maintenance of such a large machine. Intel makes an error correction chip that suits this machine well, and also makes it easy and attractive to use a scheme we thought about using in cosmic cube: two processors per node, one to run user programs and the other to handle communication. The design will probably use an 80286/80287 for the task processor and an 80186 for the communication processor.

This structure makes a very attractive homogeneous machine node, and solves a number of problems we have in (1) allowing multiple users to share the 10-cube, and (2) overhead in context switching in the 8086. Since the communication operating system would reside almost entirely in the communication processor, and the communication processor would run no user code, protection is easily achieved.

The most difficult part of this design is to find an efficient way to implement the communication channels. We are presently considering two approaches. One is to use a number of ethernet controller chips as channels, an approach that is overkill, but these are the only chips we can find that are adapted to the task. To keep the size of the node within reason, we may reduce the number of channels to 4 plus an optional spare for corner or secondary storage communication, and revert to a cube connected cycle of 4 nodes that present 8 channels to be connected in an 8-cube. Message communication performance of this connection plan is almost as good as a Boolean 10-cube. The other possible approach, but which probably contains too much risk, is to use the custom FIBT communication channel chip discussed in Section 4.4. Design and performance verification of this chip was recently completed, but there are many unknowns in putting this chip into small scale production.

Plans to build a J-11 based 10-cube machine in collaboration with DEC are currently held in suspense due to delays in the production and availability of the J-11.

2.3 Mosaic Processor

Chris Lutz, Don Speck, Steve Rabin, Pete Hunter, Chuck Seitz

2.3.1 Overview

Design of the mosaic processor was completed in December 1982, and first silicon (Figure 1) received 9 February 1983 functioned correctly to within one minor layout error in the ports discovered by simulation after the chips were sent to fabrication. Chips assembled at 4 micron feature size run at a 7 MHz clock rate. This 16-bit processor is only 1436 by 1440 lambda (without the pad frame), about 1/8th of a 6 mm square chip at 3 micron feature size. Mosaic includes 4 bidirectional communication channels, 16 registers, all common logical and add/sub instructions, bit and nibble shifts and rotates, NZCV flags, refresh address generation, 8 addressing modes, and a very efficient instruction set. This processor element is the basis of the mosaic-mesh and mosaic-tree experiments outlined above.

Needless to say, there were many smiles to be seen here on 10 February 83.

2.3.2 Design and Layout

Logical design, microcode, and simulation of a new tree machine processor, now named "mosaic," was completed by Chris Lutz between January and June 1982. The design has several unusual features. The clock cycle and microcode cycle is also the primary storage cycle. A single instruction execution may reference storage as many as 5 times, e.g. instruction, immediate data (address), operand, write result, refresh cycle. The instruction execution on this machine is pipelined, and instructions are prefetched.

The mosaic processor uses a two-phase clock driven directly onto the chip at a voltage that is allowed to exceed Vdd. Most control signals are produced by clock-and bootstrap drivers whose positive output voltage tracks that of the primary clock from which it originates. However, the clock load presented by the processor is only about 2 pf, much less than that of the bonding pad and wires. We believe that this "hot clock" and clock-and technique is extremely valuable, not only for achieving good performance and layout density, but also for making the performance of the processor much less dependent on the large variations in depletion threshold voltage between MOSIS runs.

The processor was laid out and subjected to extensive verification efforts between July and December 1982. The layout required approximately 12 person-months from 4 people, and consumed less then 100 CPU hours of VAX11/780 time. The entire layout is expressed in 10,000 lines of Earl code, a layout language with constraint solving. Critical timing paths were checked with SPICE simulation, and predicted clock period (rate) for lambda = 2 microns was 155 ns (6.5 MHz), limited by the rather conservative control PLA.

As can be seen from Figure 1, the layout is extremely regular in floorplan, but highly convoluted in some of the cell designs. This use of "Boston" geometry, that Earl supports in several ways, is estimated by its advocates

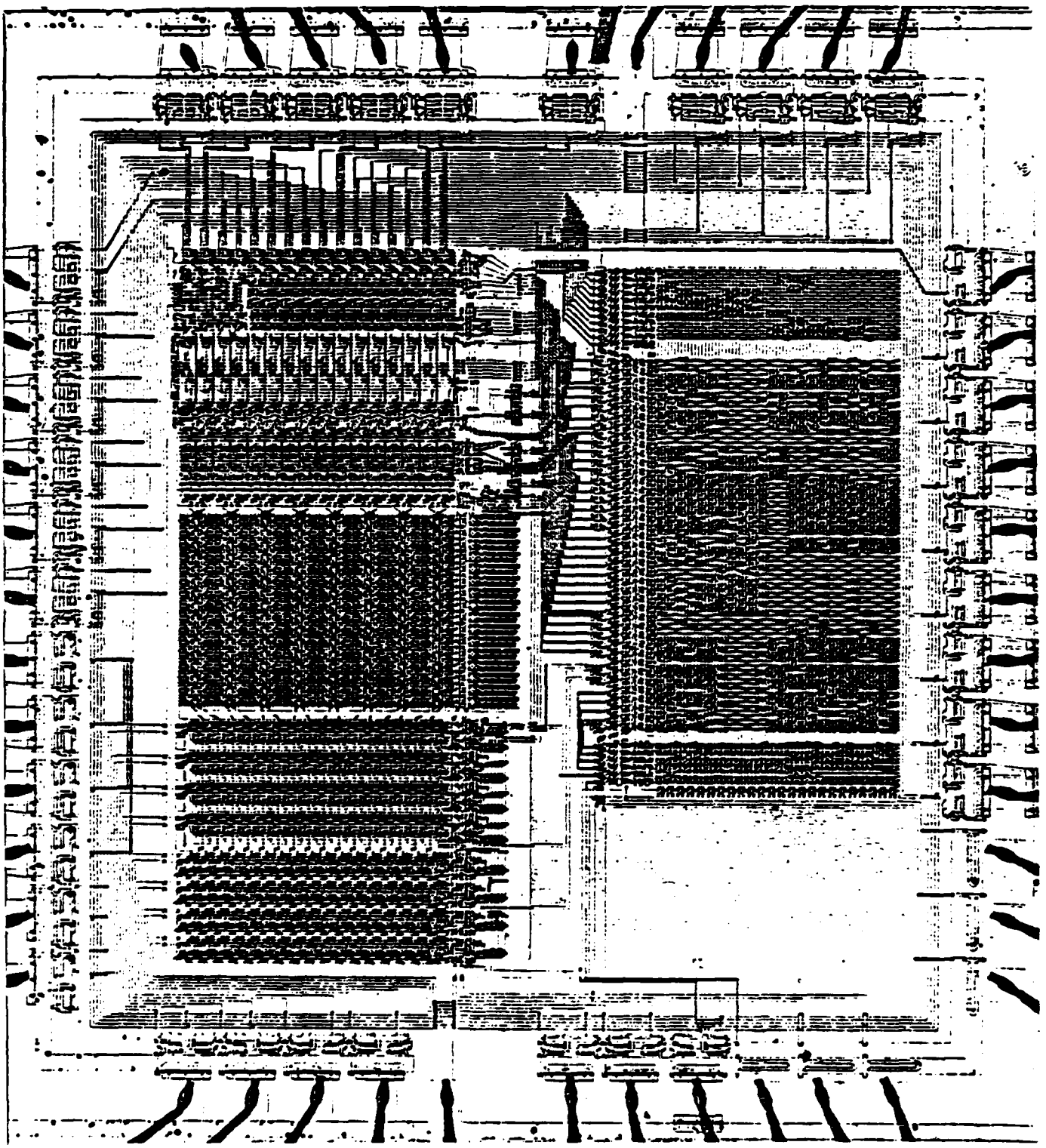

Figure 1. Prototype Mosaic Processor

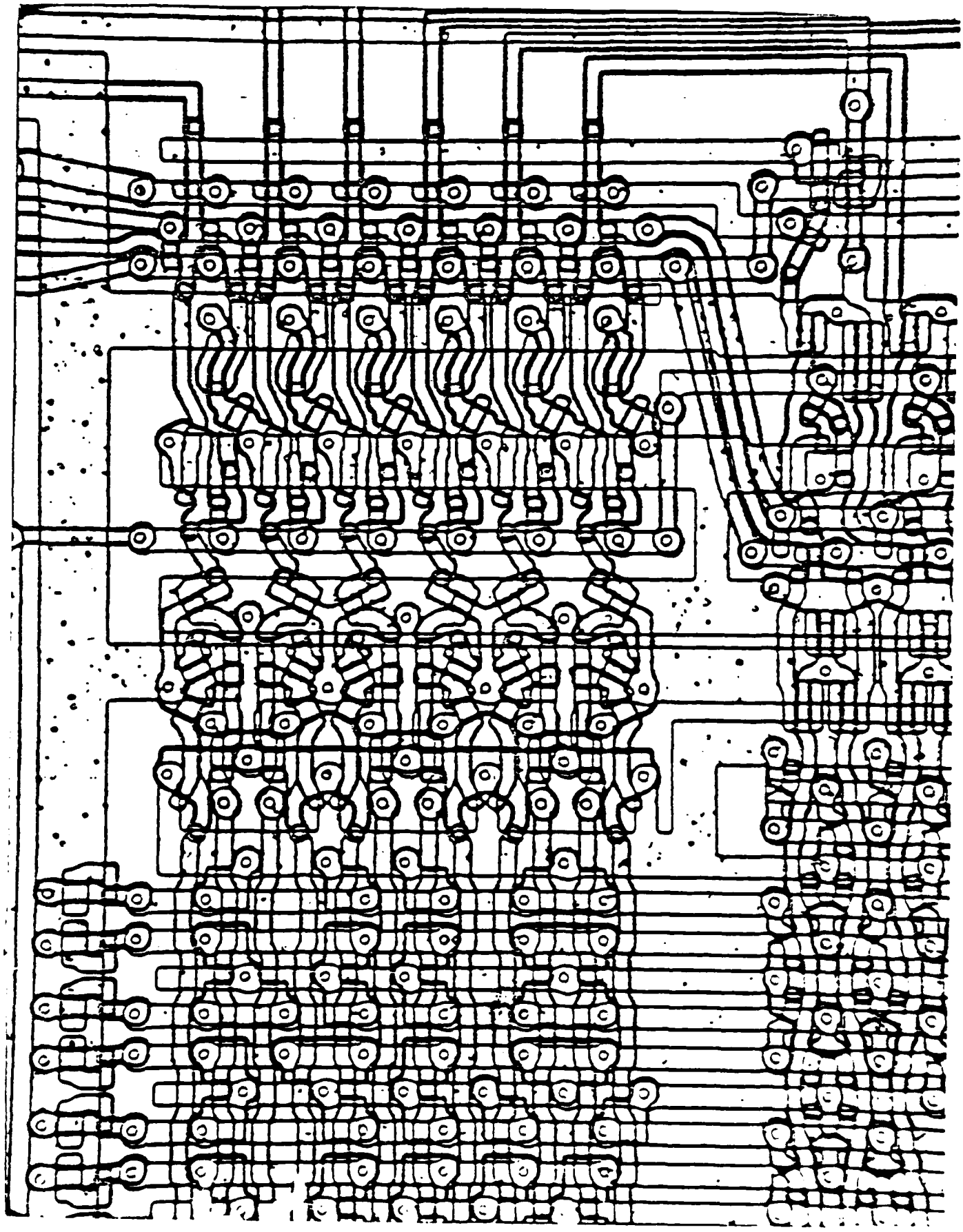

Figure 2. Illustration of use of Boston Geometry in Mosaic Processor

to have reduced the area of this layout by as much as 25% over Manhattan geometry and perhaps 10% over geometry that permits 45 degree lines. The artist responsible for the most highly repeated and most round looking sections of the design is Don Speck, a Caltech undergraduate. As can be seen from the photomicrograph of Figure 2 of the upper left corner of the control PLA, MOSIS does a very good job in scan converting this volumous and strange CIF to masks.

2.3.3 Verification Efforts

Steve Rabin, another Caltech undergraduate, served as quality control department in the final 3 months of the layout effort, and is most of all responsible for first silicon being able to execute programs.

The tree machine processor was extensively simulated using the ternary switch level simulator MOSSIM II [Schuster, Bryant 5033:TR:82]. Two means were used to generate switch level network descriptions of the tree machine processor: (1) Hand coding in a network description language from working schematics, and (2) Raster extraction of layout artwork (CIF) using the Terman/Baker algorithms [C N Baker and C Terman, "Tools for Verifying Integrated Circuit Designs," Lambda 1980, 4th Qtr. pp. 22-30, [Hedges private comm.].

Initial attempts at simulation prior to the completion of the layout were performed using a circuit description hand coded from the most recent schematics. The controller was represented as functional blocks, while the datapath was simulated at the transistor level. Primary storage was also represented at the functional level to allow simulated execution of programs.

Once the layout of the datapath was complete, graph isomorphism was established between the network descriptions and the extracted descriptions, using a new program in the suite of MOSSIM programs called MOSDIF [Schuster 83 in prep.]. Details of the full layout implementation, such as the order of inputs to nand gates, the number of inverting stages in pad amplifiers, parallel transistors, and the difficulty of keeping the hand coded schematic up to date caused us to abandon it as soon as the layout was complete, concentrating instead upon simulation of the description obtained by extraction.

Design iterations consisted of layout, circuit extraction and simulation. Several problems were found in the processor and in the layout that, while minor and easy to fix, would have made later testing very unpleasant. Although MOSSIM II proved to be remarkably free of bugs and quirks, not even MOSSIM II emerged completely unscathed — the combination of the pseudo-unit timing delay model and a sufficiently large decoder revealed an obscure bug in the simulator's event list update. This bug was fixed quickly. The authors and caretakers of the design tools we use were extremely helpful and responsive to our needs.

The simulation process was the origin of a set of functional test programs for mosaic. An new assembler was written in MAINSAIL. A small collection of test programs were written in the prototype assembly language, and their execution was simulated by MOSSIM II working from extracted layout and a

functional block simulation of main memory. The execution of the test programs revealed a few additional layout problems that were immediately fixed. These programs were also used later to test the processor.

Verification efforts used lots of computer time. CIT-VAX (VAX11/780 running Unix) accounts chuck, speck, and stever managed together to use 185 hours of CPU time in December 1982, over 6 hours per day, most of which was for extracting, plotting, and running MOSSIM.

2.3.4 Test Results

Two versions of mosaic with a pad frame wrapped around it, one assembled at lambda = 2 microns and one at lambda = 2.5 microns, were sent to MOSIS on 6 January 1983 and returned on 9 February 1983, an excellent 34 day turnaround time. The chip was tested in a simple test fixture by driving it with clocks and a reset signal and allowing it to execute code from storage. We knew it was going to work and didn't want to mess around with tedious systematic testing. And they worked!

In subsequent weeks additional test programs have been coded resulting in the discovery of two microcode problems in the prototype. Neither is critical to the functioning of the processor. Test coverage is now regarded as exhaustive.

The "LSSD" scan path between the controller and the data-path has been of no use in prototype testing. This omission is partially due to shortsightedness in the scan path design (data path control signals are not static during the period in which new data is scanned in). The storage interface has been far more useful in verifying correct behavior of the processor. In future implementations of this machine the key to processor design verification testing will be the observability of the processor/memory interface. The LSSD path may still prove to be useful for production testing.

Because of the unusual circuit and layout structures, the advantages of producing a high-yielding system part, and the desire to provide interesting reports to our friend Ms Mosis, testing of these prototype chips has been extensive. We report for the sake of completeness some data that may not be of general interest.

The initial tests were performed with the processors assembled at lambda = 2.5 microns.

All 8 processors worked correctly with a loop test program. (0: JUMP #0)

A more sophisticated test register test program (generate Fibonacci numbers, cycle them through all 16 registers, write result out to memory) worked correctly on 5 of the 8 processor chips at clock frequencies between 1 MHz and 4.6-5.0 MHz. One of the curious and still unsolved mysteries turned up at this point, that these chips will NOT work correctly at clock rates much below 1 MHz! Imagine how much trouble we would have been in if we had tried to test them at low speed!

Then some speed tests were performed with code that executed the worst-case ALU operation (adding 1 and -1, then looking at the V flag). Results:

Chip #	Maximum Speed	Maximum Phi2	Measurements done
#1	5.2 MHz	0.3μs	
#7	5.2 MHz	0.3μs	
#8	5.2 MHz	0.3μs	
#2	4.8 MHz	0.4μs	
#6	4.6 MHz	0.5μs	R1 bad on this chip
#4	4.5 MHz	0.6μs.	

These tests also included experimenting with the length of ph1 and ph2, keeping cycle time constant, and showed that ph1 could be lengthened or shortened over a 10-15ns range without affecting the maximum speed. This characteristic of the length of the clock cycle being more important than the duty cycle implied that the controller is the limiting factor, because it uses the full clock period, while the ALU is precharged on ph2 and operates on ph1.

At this point some effort was expended in looking through a microscope trying to find the reason for 3 of the 8 processors having faults. Three of the processors failed at least one diagnostic program, all in different ways.

One chip had shown problems in reading and writing register R1. The results were very speed-dependent, yielding all 1's at high speeds and hex 1800 at low speeds. After determining which register was R1 (the ninth from the left), we examined the offending bits and found that the poly had bridged. This shorted two pullups to neighboring pullups and to the read and write lines. The bad area was about 60μm across.

Another chip produced hex FC7F on the data lines on the register diagnostics. The microscope revealed that most of the poly was missing in ALU slice 8, and some in slices 7 and 9. The area of devastation was about 140μm across. The bug also produced opens in bus wires 8 and 9 where they pass through the ALU on poly. This observation easily explains this chip's problem.

The third chip failed to clear its PC on reset. It still did run 0: jump 0. We were unable to find the cause. There was a large piece of foreign matter lying on the reset code in the controller. Blowing on the chip made it go away. We don't think that was the problem.

There was a design rule violation, a one lambda poly-poly spacing discovered in the central routing cell after the chip was sent to fabrication. Checking with the microscope showed that with the chips assembled at lambda = 2 microns, this mistake was close to a short, and so the early testing was all done with lambda = 2.5 micron chips which were all examined and showed no shorts. Next we turned to testing the lambda = 2 micron chips, which turned out not to have shorts after all. Speed tests on six of the 4μm chips:

Chip #	Maximum speed	Maximum Phi2 time	(Chips tested with no warm-up time)
12	7.3 MHz	0.2µs	
9	7.0 MHz	0.2µs	D13 bond wire broken
10	6.9 MHz	0.3µs	
11	6.9 MHz	0.2µs	
14	6.8 MHz	0.2µs	
13	6.7 MHz	0.2µs	
15	-------	-----	Fails to reset
16	?	?	Cannot locate this chip.

Power supply current	120-130 mA	(Vdd = 5 v)
Power	625 mW	(about half pad drivers)
Cycle time	140-150 ns	(Vdd = 5 v, and 8 volt clocks)
Maximum Phi2 time	200-300 ns.	

These speeds were amazingly close to the figures calculated and predicted by SPICE simulation.

There were also several experiments done in varying the voltage of the "hot clocks" and Vdd. The speeds reported are at the nominal voltages of 8 and 5, respectively, but the hot clock discipline includes so much tolerance that the chip still functions at about 70% of nominal speed and less than 50% power at a clock voltage of 5 volts and Vdd of only 3 volts.

We are still uncertain why the processor fails when ph2 is longer than a few hundred nanoseconds. Our guess is that it is capacitive coupling to a control line. Some of the clock-and drivers leave their outputs floating low when deselected. Perhaps it's in the registers or (cringe) the bus.

2.3.5 Improvements and Continuing Efforts

As previously reported, we are planning to replace the present static control PLA with a precharge design that will increase speed and reduce power, but was considered a potential risk in this first implementation. We have decided to couple this change with other work currently in progress to enhance the scalability of the layout, which is currently correct only at 4-5 micron feature size. The main problems with scaling the artwork to 2-3 micron feature size are (1) power and voltage drop considerations in the controller, which will be corrected by using the precharge design, and (2) some routing and bonding problems in the pad frame.

A version 2 microcode has been written and simulated to be a better fit for the folding possibilities offered by the precharge version of the control PLA. This change in the control PLA also makes it desirable to change some of the routing plan between the controller and datapath, saving about 70 lambda in the vertical dimension. The assembler is being revised to assemble code for the version 2 microcode. We expect to accomplish and iterate these improvements over the summer while developing the storage element.

We expect the result of these improvements to be a small decrease in area and an increase in speed so that processors fabricated at 3 micron feature size will operate sufficiently comfortably in excess of 10 MHz that we can use a 10 MHz system clock for mosaic-mesh and mosaic-tree.

2.3.6 Floating Point Routines

Many of the applications planned for mosaic are heavy on floating point computations. Although the processor does not support floating point directly in either hardware or microcode (to do so would vastly increase the complexity of the processor), the instruction set lends itself well to software floating-point. Chris Lutz has written an elegant floating point package for the processor.

Floating-point quantities are represented in three words: two words (32 bits) of normalized mantissa, and one word containing a mantissa sign bit and 15 bits of two's complement exponent. Thus a range of greater than 10 raised to $\pm$ 5000 can be represented.

The floating point package contains add, subtract, multiply, and divide subroutines that yield 32-bit rounded results in all cases that the result is representable. Underflow and overflow can be trapped by the user. The package is 200 words long. Typical execution times (in processor microcycles and in μs assuming a 10 MHz clock) are:

Add and subtract:	180 microcycles;	18 μs
Multiply:	870 microcycles;	87 μs
Divide:	960 microcycles;	96 μs.

The package can be modified to handle 48-bit mantissas (4 words per floating-point quantity) by insertion of 28 additional words. Execution times will then be increased by 10% for add and subtract, and by 70% for multiply and divide.

2.3.7 Storage Element

The dynamic RAM test chip designs to provide on-chip storage for mosaic have been returned from MOSIS, but have a nearly fatal design error in the decoder. Nevertheless, we have enough experience through student designs with several compact 3 transistor dynamic RAM cells, and through test structures with the clock-and drivers, to proceed with a storage element that employs bootstrap address line drivers. The current design adds a ph1x clock at the end of the ph1 period.

Meanwhile, partly from experience with the cosmic cube, we have become concerned about the soft error consequences of building systems with very large amounts of dynamic storage. Very large ensembles of mosaic processors with on-chip memory may require error correction in order that the mean time between failure be more than a few minutes. Whether error correction is needed depends on the number of processors in the ensemble, the amount of memory per processor, and failure statistics in the 3-transistor dynamic RAM cell we are using.

Since all commercial dynamic memory currently uses 1-transistor cells, statistics must be obtained by fabricating and testing experimental versions of the memory. Several properties of the mosaic storage element make it less likely that error correction will ever be needed: the 3-transistor cell has little diffusion area on the storage node, making it difficult for photoelectrons from alpha radiation to discharge the node; the 3-transistor

cell has a much higher noise margin than a 1-transistor cell since there are no sense amps and sharing of large read line capacitance with small cell capacitance; and the high refresh rate guaranteed by the processor (at least one of every 7 microcycles is a refresh cycle) reduces alpha susceptibility.

In the event that error correction is needed, however, Chris Lutz has been investigating some of the possibilities. The error correction problem in mosaic mesh or mosaic tree is different from that in current commercial computers in several ways: (1) The storage is synchronous with the processor and must be very fast (10 MHz). Since error correction is in series with memory access, it must be comparably fast in order not to slow the entire system down. (2) The error correction is entirely on-chip, so LSI MOS design techniques (taking into account the large cost of communication and small cost of regular structures) apply. (3) Correction of hard errors is useful: to prevent immediate system failure on hard error; and to increase chip yield at the expense of soft error correction capability. If both these goals are sacrificed, then properties applying only to soft errors can be exploited. In particular, soft errors in the mosaic storage may be confined almost exclusively to discharging of storage nodes due to alpha radiation. There is no mechanism for erroneously charging a storage node. The error correcting code can take this asymmetry into account.

Codes that need correct only single errors of 1->0 transitions are at best only one bit shorter than the (reduced) Hamming code. Furthermore, the known examples of these codes are far more analytically and computationally intractible than the Hamming code, with two exceptions: The code with only the two codewords [0,0] and [1,1] is trivial to encode and decode. (To decode, simply find the logical OR of the bits.) Its overhead of a factor or two in storage requirements is excessive, however, unless both bits can be efficiently built into one storage cell, and in such a way that an alpha particle cannot discharge both bits. The other exception is to simplify any syndrome decoding scheme (e.g. any linear code) by forming not the exclusive-OR of the error pattern with the word to be corrected, but rather simply the OR. OR can be performed more quickly, in less space, and with less power than exclusive-OR.

Most commercial computers correct main storage with a reduced Hamming code (a perfect, single-error correcting code which is truncated to give the the word size of the particular machine). Strong possibilities for the mosaic storage element are the (21,16), the (38,32), and the (71,64) reduced Hamming codes. (The first number in parentheses is the length of the codeword, the second is the number of data bits represented.) The Hamming code suffers, however, from the need to compute the many-input exclusive-OR of approximately half of the codeword bits, roughly evenly distributed across the word. This computation yields the syndrome (a unique indication of the error location) which must be distributed to comparitors at every codeword bit. Although not expensive in hardware, this is a very lengthy computation.

A convolutional code can solve the problems of many-input exclusive-ORs and long communications paths. A convolutional code is a linear code in which a corrected data bit is a function of only the codeword bits in some neighborhood around it. For example, a code exists which requires 1 check bit for every 3 data bits, and computation of the syndrome requires exclusive-ORing

only 7 bits out of a neighborhood of 13 codeword bits. Each syndrome bit is used only in its neighborhood of 12 data bits. Arbitrary word lengths can be obtained simply by repeating a cell representing a block of 3 data bits.

2.3.8 Mosaic Mesh and Mosaic Tree

As sufficient numbers of prototype mosaic chips are received from MOSIS, we will be assembling a number of prototype mosaic tree and mosaic mesh machines using commercial (InMOS 4K x 4) storage chips.

Work is also underway in the logistics of wafer testing mosaic mesh and mosaic tree elements with on-chip storage. A wafer stepper probe station is set up and is being interfaced to our computers.

2.4 Super Mesh

Wen-king Su, Chuck Seitz

2.4.1 Super Mesh (Seitz)

Guided largely by seeing how the physics people have been programming regular applications on the cosmic cube, we have recently started working on the design of an SIMD machine of quite small grain size, about 1 MSL. It seemed very wasteful in cosmic cube that the same program was loaded into all of the nodes of this MIMD machine, even though the execution temporarily follows different branches in different machines. We have always rationalized this duplication of code as necessary in face of the cost of broadcasting control. However, if an efficient broadcasting mechanism could be devised, an SIMD machine of quite small node size could start to reach into the range of a factor of 1000 in cost/performance over conventional uniprocessors.

An answer to this dilemma is serialism. As was pointed out many years ago in studies of the Illiac IV, serial nodes in parallel computers are inherently efficient. Where n-bit parallel arithmetic in time T requires n full adders of carry delay T/n, n-bit serial arithmetic in time nT requires only 1 full adder with delay T. The ratio in power and cost between these structures is n^2, while the ratio in performance is only n. Illiac would have been a more cost effective machine with 64 times as many nodes each 64 times slower, and 64^2 times cheaper. Slower serial nodes also simplify the instruction issuing computer and the broadcast of control, indeed allow and encourage this broadcast to occur serially. The physical structure of the clock and microinstruction distribution require that the interconnection plan be a 1-, 2-, or 3-dimensional mesh.

A VLSI engine that follows from this reasoning, dubbed super mesh (SM), actually resembles a computational or systolic array, except that instead of the algorithm being built into the node, the node operates on the contents of its registers in response to the instruction sequence broadcast through the ensemble. We expect that the existing body of algorithms developed for computational arrays, and the programmability of this implementation, makes super mesh a good approach toward the engineering of computational arrays.

A strawman design of the node includes registers for 16 floating point numbers, and a carry-save technique for the mantissa multiplication. Because of the uniformly short paths through combinational logic, such a machine should be able to operate at a bit rate in excess of 20 MHz, which for a 32-bit mantissa translates into about 5 microseconds per floating point multiply.

One can do an analysis for optimal returns in speed for area in carry save multiply algorithms with m adder elements for n bit words. The analysis suggests that m=n is optimal, but the function is curious, and of course in the continuous approximation has lots of discontinuities. For m<n the area time product is about constant to within non-integral ratios of n/m. For m>n you have the curious situation that for kn < m =< (k+1)n the clock period is (k+1) times what it is for m=n, so for example where m=2n the number of clock periods required to do the multiply is reduced to half, the period is twice as large, so there is obviously no winning even in absolute speed (except for a 2nd order term) along this m>n route. The reason we regard m=n as optimal rather than some m<n is that the multiplier is only a fraction of the area of the system, and so making it twice as big (using actual numbers) from about 0.1 to 0.2 MSL and twice as fast does not greatly impact the total area, about 1 MSL, required.

This strawman machine turns out to have too much arithmetic performance and too few registers to be a good fit to real problems. It very well reveals a problem inherent in the computational array model, in that for more than one-dimensional arrays, the time complexity of serial loading exceeds that of computing. Although it may seem counterintuitive, a realistic design will have to include much more storage relative to the arithmetic capability, in effect to share the arithmetic capability amongst more operands, and to permit a broader class of applications. Since registers on MOSIS chips are not very cost effective simply as storage, we are now looking at ways to back up supermesh with a large secondary RAM store using high-density commercial RAM chips.

Although supermesh is still in the early stages of simulation and design, we can project conservatively that a small system of 64 supermesh chips (1024 nodes) plus 256 storage chips, will achieve present supercomputer performance for "systolic" problems in which the number of iterations is large enough to outweigh the loading time.

3. Concurrent Computation

3.1 Concurrent Algorithms

Lennart Johnsson

During the past six months a few algorithms for the preconditioned conjugate gradient method have been devised for binary trees and Boolean n-cubes. Preconditioning adds a O(N) time complexity term to the conjugate gradient method for full matrices. For sparse matrices either the space or the time complexity, or both can be improved. Preconditioning implies that linear recurrances have to be solved in each step of the conjugate gradient method. With matrices in band form the devised algorithms still run in O(N) time, but with matrices in bordered block diagonal form the time complexity can be reduced to O(logN+M) where M is the block size, [Johnsson 83].

A new course on concurrent algorithms was offered during the winter quarter. The focus in the course is on highly concurrent algorithms, their communication needs, distributed data structures, and the mapping of algorithms onto ensembles of sparsely interconnected processors.

[Johnsson 83] S.L. Johnsson, "Highly concurrent algorithms for solving linear systems of equations". Presented at "Elliptic Problem Solvers", Monterey, CA. January 1983. To appear in Elliptic Problem Solvers II, Academic Press.

3.2 Tree Machine Software

Pey-yun Peggy Li, Lennart Johnsson

3.2.1 Loader

Three improved downloading algorithms have been analyzed, implemented and tested on the tree machine simulator. These algorithms speed up the time to load the node types from O(N) to at best O(log N). For a logical tree with totally N nodes of fanout f, the algorithm that takes the description of the logical tree as input achieves the best performance, O(log N) in all cases. The tree processors do the conversion from a fanout f tree to a binary tree. The other two algorithms take the description of the converted binary tree as input have a worst case performance O(sqrt(N)/f) and O(N**(1/log f)), [Li, Johnsson 83]

3.2.2 Mapping of Logic Trees onto Fixed Binary Trees

1. Balancing a binary tree: three algorithms, AVL height balancing, bottom-up balancing and an in-tree top-down algorithm have been analyzed. The algorithm which has to be implemented in the host is not desirable, because it always increases the length of the node type sequence which will be fed into the tree at downloading time. The in-tree top-down algorithm has the best performance in terms of the total increase of the communication cost. But, the algorithm complexity is too high to be satisfactory, namely, O(N).

2. Timesharing of nodes: A simple mapping algorithm has been designed to map a bigger binary tree onto a smaller tree machine. Each tree machine processor will be timeshared by more than one node in the problem tree. The adjacency relation, i.e. father-son relation, will be preserved after mapping. In other words, the adjacent nodes in the converted binary tree will be mapped either onto one processor or two connected processors in the tree machine. Hence, the mapping algorithm never increases the communication path.

3.2.3 Scheduler

A scheduler is being implemented in the tree machine assembly language. The scheduler keeps track of the states of processes, interprets the incoming and outgoing messages and makes corresponding state transitions and context switches. The most important task of the scheduler is to handle the input and output interrupts, then place the message into the proper port of the proper process and evoke the process which has been suspended on waiting for that message.

[Li, Johnsson 83] Peggy Li and Lennart Johnsson, "The Tree Machine, Strategies For Reducing the Program Loading Time", Computer Science, Caltech, February 1983

3.3 Residue Arithmetic

Chao-lin Chiang, Lennart Johnsson

The computation speed of arithmetic based on the residue number system is potentially much greater than the speed using the binary number system. There is no long carry chain in the residue number system.

The maximum speed-up of the multiplication of two N bit integers is O(N/logN). This limit can be reached only if the individual residue arithmetic units are implemented efficiently. Traditionally, table look-up methods are used to implement a residue arithmetic unit. Table look-up methods are practically limited to computations with numbers represented by less than 5 bits.

Three different types of residue multipliers are being designed: simple table look-up, a combination of tables and adders, and an array of adders. These multipliers are now being implemented in NMOS technology with 4 micron feature size. Simulation results indicates that for a number represented by less than 4 bits the table look-up method is preferable. The second alternative offers advantages in the range 5 to 6 bits, and the third alternative is competitive when more than 6 bits are required. These results are subject to change with the technology.

As MOS technology is scaled down, the wire delay will dominate in the PLA or ROM structures (due to the long polysilicon wires). Hence, the methods that use tables do not scale well. The third alternative (array adder) will be increasingly competitive in lower bit ranges.

With proper choice of bases, one can construct a residue number system of required range. The following speed-up figures are derived for residue arithmetic compared to binary arithmetic using a combinational multiplier.

BASE	RANGE(bit=N) (combinational multiplier)	SPEED-UP RATIO
5,7,9,11,13,16	20	7.5
5,7,9,11,13 32,17,31,29	35	7.5
7,11,13, 32,17,31,29 27,25,23,19	47	10.0
11,13 17,19,23,25,27 29,31 37,41,43,47,53 59,61,49,64	90	10.6
	:	:
		(N/logN)

3.4 "The Mathematics of the Black Box" or, a General Proof Rule for Procedures

Alain Martin

In all (hardware or software) design disciplines, the concept of "black box" plays a crucial role: after a component has been designed for achieving a certain net-effect, it can be replicated and used without knowing its internal structure, but knowing only what it does.

The programming concept of "procedure" is a good formalism to study this problem in general. There are two main issues. The first one is the "parameter mechanism" by which an instance of a black box is linked to a certain environment. The second one is a partial specification issue. A procedure is only a partial semantic function: it has been designed to establish a certain post condition, but unlike other programming constructs, it is not known how it establishes any postcondition. In [Martin 83], a proof rule for using procedure is proposed that we believe to be more general and easier to use than other existing ones. Assuming that it has been proved that the body of a procedure establishes a certain postcondition I for a certain precondition J, the rule permitting to determine under which conditions a certain procedure call establishes the postcondition E is based on finding in adaptation A such that $A \wedge I \Rightarrow E'$. (E' is derived from E by some substitution of parameter variables.)

[Martin 83] Martin, Alain J. A general proof rule for procedures in predicate transformer semantics. Caltech techical report 5075:TR:83. Feb. 1983. (submitted for publication)

3.5 FFTs and Posets on the Cosmic Cube

Michael Newton, Chuck Seitz

The hypercube interconnection plan makes the cosmic cube particularly nice for certain forms of computation.

The FFT is one example. The butterfly operations involved in computing the Fast Fourier Transform are essentially the same pattern as the interconnection pattern on a hypercube. Thus, when data is already present in the correct nodes in the Hypercube, it is possible to run a FFT in log(n) time, [Newton, 5057:DF:82].

Poset problems are another example. Consider a graph G, with vertices V = {V1, V2, V3, . . . Vn} and edges E = {V1V2, V1V3, ...V1Vn, V2V3 . . . Vn-1Vn} and a function f:E->N, N being the natural numbers. We wish to form a closed path connecting all the vertices once and only once, e.g., create a path Vi1Vi2Vi3....Vin such that sum of f(i) on the path is minimal/maximal. This problem, usually referred to as the Traveling Salesman Problem, grows exponentially in difficulty with the length of the input string (roughly the number of cities), and is of a class of problems generally referred to as NP complete. This problem can also be thought of as the minimization of the cost function of a full sequence of the non-commutative elements of V. The elements, however, are associative, so partial results (of the form Vi,j Vi,j+1...Vi,j+k) are useful in more than one calculation. These partial results can be generated in the form of a Complete Boolean Partially Ordered Set (poset). As shown in [Newton, 5037:DF:83], posets of this form are isomorphic to a Boolean Hypercube. Using this isomorphism it is possible to write algorithms to solve this and similar problems of size smaller than the dimension of the machine in n^3 time. Even for problems larger than the size of the machine the worst part of the exponential growth in time is removed.

3.6 Circuit Simulation on the Cosmic Cube

Sven Mattison, Lennart Johnsson, Chuck Seitz

We have started to investigate the possibility of writing a circuit simulator similar to SPICE for the cosmic cube, for two reasons. One reason is that SPICE uses lots of time on our computers as well as on many people's supercomputers, and such a program would provide a more economical way to doing these simulations. The second reason is that circuit simulation is an excellent paradigm of problems in which the communication graph, while fixed, is not so regular as in the problems done by our collaborators in the sciences at Caltech.

There are massive concurrencies available in the model evaluation phase of such a circuit simulator, and we estimate that a concurrent approach can reduce this time from about 80% to less than 20% of the running time, thus perhaps allowing more elaborate models to be used if desired. The solution of the resulting linear system benefits from recent results in solving sparse matrix equations, [Johnsson, 4087:DF:80, 5040:TR:82]. The efficiency of this computation depends on the way in which the circuit variables are mapped into physical processors. An efficient mapping need be done only

once, and approaches based on annealing and on locality in extracted networks are both being considered. A prototype program is Pascal, written to extend easily to a concurrent system, has been written to experiment with the linear system solution and time step techniques.

4. VLSI Design

4.1 A Self-timed Circuit for Distributed Mutual Exclusion

Alain Martin, Mike Schuster

As an introduction to the study of self-timed design methods, a self-timed circuit for mutual exclusion has been designed. If N independent processes can be simultaneously a candidate for the exclusive access of a shared resource, the mutual exclusion circuit sees to it that the simultaneous candidates are granted the resource in some order. Using a solution proposed in [Martin 82], two approaches have been taken. In the first one, the circuit is directly derived from the program. In the second one, based on the method proposed in [Snepscheut 83], a finite state machine is first derived from the program and the circuit is an implementation of the finite state machine. The two approaches lead to quite different circuits.

[Martin 82] Martin, Alain J. Distributed Mutual Exclusion Algorithms, Caltech technical report 5047:TR:82. Sept. 1982.

[Snepscheut 83] V.D. Snepscheut, Jan. "Deriving Circuits from Programs." Proc. Third Caltech Conf. on VLSI, March 1983

4.2 From Circuit to Layout - Another Approach

Tak-Kwong Ng, Lennart Johnsson

Many "chip" layouts may have identical logical behavior. It is this many-to-one mapping from layout to logic which makes the task of automatic generation of "efficient" layout very difficult. In order to generate the "best" layout for a logic design one need to evaluate every possible layout. This certainly is a large combinatorial problem. To reduce the problem complexity one may partition all possible layouts with an equivalent relation. The choice of equivalent relation depends on the evaluation criterion.

One possible evaluation criterion is the number of contacts in the layout. The equivalent relation for this evaluation criterion is defined. The development of an algorithm which computes the minimum number of contacts of any block in the partition is in progress. The development of an efficient searching algorithm for all blocks with minimum number of contacts is also in progress.

The "cross-over" problem is the problem of finding a layout for implementing a given circuit with minimum number of contacts. The complexity of this "cross-over" problem is expected to be like the graph coloring problem. There is no known polynomial bound algorithm for solving the graph coloring problem except for the one or two coloring cases. For the same reason a polynomial bound algorithm for solving the "cross-over" problem may not be found except for the zero cross-over case. It is our belief that efficient algorithms based on the branch-and-bound technique can be found for most circuits.

Other evaluation criteria under consideration are the dimensions of the minimum bounding box of a layout and the maximum wire length.

4.3 FBRT Chip

Charles Ng, Chuck Seitz

A fifo buffered receiver transmitter (FBRT) [Ng, 5055:TR:82] one-to-one communication chip for future experimental homogeneous machines was tested in March 83 and found to work correctly at a 6 MHz clock rate. The first iteration of this design tested in December 82 was only partially functional due to a ratio error.

This unusual chip employs a pair of asynchronous FIFOs, one for each direction, to provide packet storage and speed matching between a microprocessor interface on one side and a communication channel interface on the other side. The communication channel requires only one wire in each direction. Packets are sent at a bit rate determined by a clock that is distributed throughout the system, or can be generated from a crystal independently. The clock serves only as a frequency reference; its phase is arbitrary. Acknowledgements that confirm that a packet was received and there is now room for a new packet are special packets, interspersed with but with higher priority than data packets, on the opposite direction wire for the channel.

This design is highly parameterized, including word format and packet length, and several channels can be included on a single chip. It would be very nice to be able to use this chip in a future cosmic cube class of machine if the logistic problems of putting it into small scale production could be solved. Alternatively, this design may be turned over to DEC for them to produce for J-11 based homogeneous machines.

4.4 Hierarchical Router

John Y. Ngai, Chuck Seitz

Since the last report, work has been focused primarily on the following:

(1) A revision of the system's input specifications to include features that allow easier user interactive control.

(2) An modification of the routing model to handle gridless channel routing.

(3) An extension of the channel routing algorithm to handle two-dimensional routing, ie, pins on all four sides of a channel.

(4) Fine tuning of the routing heuristics involved in getting better layout.

(5) Implementation and testing of the system.

The prototype implementation is near completion, and we expect to run large real tests using it soon.

4.5 SOS Technology and Libraries

Chuck Seitz, Sven Mattison

Using primarily students in the VLSI Design Laboratory course, efforts to develop our design capability in CMOS/SOS are continuing well. The first MOSIS CMOS/SOS run done in November 82 included over 40 CMOS/SOS projects, with all but 7 students meeting this early fabrication deadline. About half of the students elected to do their second project in CMOS/SOS instead of nMOS, particularly those with prior industrial experience, who pointed out that almost all new designs done in their companies were in CMOS.

Sven Mattison has developed and is continuing to refine a set of SPICE parameters for CMOS/SOS that seem to produce accurate simulations.

Another project carried out in the winter quarter by the class is an extensive library of nMOS and CMOS/SOS cells written in Earl, and often very well parameterized. This cell library is now being distributed along with Earl.

California Institute of Technology
Computer Science, 256-80
Pasadena, California 91125

ARPA Technical Memos and Technical Reports
together with selected other Caltech reports on VLSI topics
March 1983

Available from the Computer Science Department Library
Prices include postage
and help to defray our printing and mailing costs.

+++

5074:TR:83 "Robust Sentence Analysis and Habitability," Trawick, David J., Ph.D. Thesis, February 1983

5073:TR:83 "Automated Performance Optimization of Custom Integrated Circuits," Trimberger, Stephen, Ph.D. Thesis, March 1983

5068:TM:83 "A Hierarchical Simulator Based on Formal Semantics," Proc. Third Caltech Conf. on VLSI, p. 207-223, March 1983; Chen, Marina and Carver Mead [$1.00]

5065:TR:82 "Switch Level Model & Simulator for MOS Digital Systems," December 1982; Bryant, Randal E. [$3.00]

5059:TM:82 "A Comparison of MOS PLAs," December 1982; Trimberger, Stephen [$2.00]

5055:TR:82 "FIFO Buffering Transceiver: A Communication Chip Set for Multiprocessor Systems," MS Thesis, December 1982; Ng, Charles H. [$5.00]

5052:TR:82 "Submicron Systems Architecture," submitted to ARPA October 1982; Semiannual Technical Report [$4.00]

5049:TR:82 "Distributed Mutual Exclusion Algorithms," submitted for publication, AJM 31, September, 1982; Martin, Alain [$3.00]

5047:TR:82 "The Torus: An Exercise in Constructing a Processing Surface," Proc. 2nd Caltech Conference on VLSI, Caltech, Pasadena CA, January 1981; Martin, Alain [$3.00]

5046:TR:82 "An Axiomatic Definition of Synchronization Primitives," Acta Informatica 16, pp. 219-235 (1981); Martin, Alain [$3.00]

5045:TM:82 "A Distributed Implementation Method for Parallel Programming," Proc. Information Processing '80, S. H. Lavington, (ed.); Martin, Alain [$3.00]

5044:TR:82 "Hierarchical Nets: A Structured Petri Net Approach to Concurrency," September, 1982; Choo, Young-Il [$10.00]

5043:TM:82 "A Formal Derivation of Array Implementations of FFT Algorithms," Proc. USC Workshop on VLSI & Modern Signal Processing, (sponsored by ONR) Nov. 1982, to be published by Prentice-Hall; Johnsson, Lennart and Danny Cohen [$3.00]

5042:TR:82 "Concurrent Algorithms as Space Time Recursion Equations," September, 1982; Chen, Marina and Carver Mead [$4.00]

5040:TR:82 "Concurrent Algorithms for the Conjugate Gradient Method," September, 1982; Johnsson, Lennart [$4.00]

5038:TM:82 "A New Channel Routing Algorithm," September, 1982; Chan, Wan S. [$4.00]

5035:TR:82 "Type Inference in a Declarationless, Object-Oriented Language," MS Thesis, June 1982; Holstege, Eric [$10.00]

5034:TR:82 "Hybrid Processing," Ph.D. Thesis, March 1982; Carroll, Chris [$12.00]

5033:TR:82 "MOSSIM II: A Switch-Level Simulator for MOS LSI User's Manual," August 1982; Schuster, Mike and Randal E. Bryant [$4.00]

5030:TM:82 "VLSI Algorithms for Doolittle's, Crout's, and Cholesky's Methods," Proc. ICCC '82, IEEE Int'l Conf. on Circuits & Computers, NY Sept. 1982, pp.372-377, IEEE Catalog No. 82CH1813-5; Johnsson, Lennart [$1.00]

5029:TM:82 "POOH User's Manual," July 1982,; Whitney, Telle [$4.00]

5027:TM:82 "Concurrent Programming," July 1982; Bryant, Randal E. and Jack B. Dennis [$3.00]

5021:TR:82 "Earl: An Integrated Circuit Design Language," MS Thesis, May 1982; Kingsley, Chris [$5.00]

5019:TR:82 "A Computational Array for the QR-Method," Proc. MIT Conference on Advanced Research in VLSI, P. Pennfield, ed., Boston, January 1982, pp.123-129; Johnsson, Lennart [$3.00]

5018:TM:82 "Filtering High Quality Text for Display on Raster Scan Devices," August 1981; Kajiya, Jim and Mike Ullner [$2.00]

5017:TM:82 "Ray Tracing Parametric Patches," May 1982; Kajiya, Jim [$2.00]

5016:TR:82 "Bristle Blocks - Scrutinized and Analyzed," June 1982; McNair, Richard, and Monroe Miller [$4.00]

5015:TR:82 "VLSI Computational Structures Applied to Fingerprint Image Analysis,"; Megdal, Barry [$4.50]

5014:TR:82 "The Extension of Object-Oriented Languages to a Homogeneous, Concurrent Architecture," Ph.D Thesis, April 1982; Lang, Charles R. Jr., [$15.00]

5012:TM:82 "Switch-Level Modeling of MOS Digital Circuits,"; Bryant, Randal [$2.00]

5005:TM:82 "Chip Assembly Tools,"; Trimberger, Steve and Chris Kinglsey [$2.00]

5004:TM:82 "Riot - A Simple Graphical Chip Assembly Tool,"; Trimberger, S. and Jim Rowson [$2.00]

5003:TM:82 "Pipelined Linear Equation Solvers & VLSI ," Proc., Microelectronics 1982, Adelaide, Australia, May 1982, pp.42-47, The Institution of Engineers, Australia, Nat'l Conf. Pub. No. 82/4; Johnsson, Lennart S. [$2.00]

5001:TR:82 "Minimum Propagation Delays in VLSI ," IEEE J. Solid State Circuits, Vol. SC-17, No. 4, August 1982, pp. 773-775; (later version of 4601:TM:81); Mead, Carver, and Martin Rem [$2.00]

5000:TR:82 "A Self-Timed Chip Set for Multiprocessor Communication," MS Thesis, February 1982; Whiting, Douglas [$6.00]

4778:TM:82 "Testing and Structured Design," Proc. International Test Conference., Cherry Hill NJ, August 1982; DeBenedictis, Erik P., and Charles L. Seitz [$2.00]

4777:TR:82 "Techniques for Testing Integrated Circuits,"; DeBenedictis, Erik P.

4724:TR:82 "Concurrent, Asynchronous Garbage Collection Among Cooperating Processors," (superceded by 5014:TR:82); Lang, Charles R. [$5.00]

4716:TM:82 "A Rectangular Area Filling Display System Architecture,"; Whelan, Dan [$4.00]

4710:TM:82 "Earl: An Integrated Circuit Design Language," (Succeeded by 5021:TR:82); Kingsley, Christopher [$5.00]

4684:TR:82 "A Characterization of Deadlock Free Resource Contentions,"; Chen, Marina, Martin Rem, and Ronald Graham [$3.00]

4682:TR:81 "Earl: An Integrated Circuit Design Language," (Succeeded by 5021:TR:82); Kingsley, C. [$3.00]

4675:TR:81 "Switching Dynamics," MS Thesis,; Lewis, Robert K. [$7.00]

4655:TR:81 "Proceedings Second Caltech Conference on Very Large Scale Integration," 19-21 January 1981; Seitz, Charles, ed. [$20.00]

4654:TR:81 "A Versatile Ethernet Interface," MS Thesis; Whelan, Dan [$12.00]

4653:TR:81 "Toward A Theorem Proving Architecture," MS Thesis; Lien, Sheue-Ling [$10.00]

4618:TM:81 "The Tree Machine Operating System,"; Li, Peggy [$5.00]

4601:TM:81 "Minimum Propagation Delays in VLSI," Proc., Second Caltech Conf. on VLSI, January 1981; Mead, Carver A. and Martin Rem [$3.00]

4600:TM:81 "A Notation for Designing Restoring Logic Circuitry," Proc., Second Caltech Conf. on VLSI, January 1981; Rem, Martin, and Carver A. Mead, (revised from 4529:TM:81) [$3.00]

4530:TR:81 "Silicon Compilation," Ph.D. Thesis; Johannsen, Dave [$18.00]

4521:TR:81 "Lambda Logic," MS Thesis; Rudin, Leonid [$8.00]

4517:TR:81 "The Serial Log Machine," MS Thesis; Li, Peggy [$7.00]

4407:TM:82 "An Experimental Composition Tool,"; Mosteller, Richard C. [$3.00]

4379:TR:81 "LAP User's Manual,"; Lang, D. (Rev. by C. Kohle and S. Trimberger 9/81) [$2.00]

4336:TR:81 "A Structured Design Methodology and Associated Software Tools,"; Trimberger, S., J. Rowson, D. Lang, and J.P. Gray [$4.00]

4334:TR:81 "An Inexpensive Multibus Color Frame Buffer,"; Whelan, Dan and R. Eskanazi [$1.00]

4332:TR:81 "RLAP, Version 1.0, A Chip Assembly Tool,"; Mosteller, R. [$3.00]

4320:TR:81 "A Hierarchical Design Rule Checker," MS Thesis; Whitney, Telle [$7.00]

4317:TR:81 "REST - A Leaf Cell Design System," MS Thesis; Mosteller, Richard C. [$10.00]

4298:TR:81 "From Geometry to Logic," MS Thesis,; Lin, Tzu-mu [$5.00]

4287:TR:81 "Computational Arrays for Band Matrix Equations,"; Johnsson, L. [$2.00]

4281:TR:81 "Combining Graphics and a Layout Language in a Single Interactive System (2nd Revision),"; Trimberger, S. [$2.00]

4270:TR:81 "FIFI Test System Preliminary User's Manual," (later version contained in 4777:TR:82); DeBenedictis, Erik [$3.00]

4204:TR:78 "A 16-Bit LSI Digital Multiplier," EE Thesis; Masumoto, R. T. [$8.00]

4191:TR:81 "Towards A Formal Treatment of VLSI Arrays," Proc., Second Caltech Conference on VLSI, Pasadena, CA, January 1981; Johnsson, Lennart S., Uri Weiser, D. Cohen, and Alan L. Davis [$4.00]

4168:TR:81 "Computational Arrays for the Discrete Fourier Transform," Proc., 22nd Computer Science Int'l. Conference, CompCon 81, San Francisco, February 1981, pp. 236-244, IEEE Catalog No. 81CH1626-1; Johnsson, Lennart S. and D. Cohen [$3.00]

4088:TR:80 "The Representation of Communication and Concurrency,"; Milne, George [$8.00]

4087:TR:80 "Gaussian Elimination of Sparse Matrices and Concurrency - A Complexity Analysis," (Succeeds 4087:DF:80); Johnsson, Lennart [$3.00]

4061:TR:80 "A Preliminary Report on the Caltech ARPA Tester Project," (later version in 4777:TR:82); DeBenedictis, Erik [$4.00]

4029:TR:80 "Structure, Placement and Modeling," MS Thesis; Segal, R. [$8.00]

4025:TM:80 "Sticks Standard Software Package,"; Segal, R. and Steve Trimberger [$2.00]

4024:TM:80 "SSP Basic Software Package (Revised),"; [$5.00]

4022:TM:80 "Comprehensive CIF Test Set (Revised),"; Trimberger, S. [$5.00]

3901:TM:78 "Hierarchical Design for VLSI,"; Rowson, J. [$3.00]

3882:TM:80 "A Chip Assembler,"; Tarolli, Gary [$2.00]

3880:TM:80 "The Proposed Sticks Standard,"; Trimberger, S. [$5.00]

3857:TM:80 "VLSI Architecture & Design," Proc., National Electronics Conference, Chicago, Oct., 1980, Vol. 34, pp. 254-259; Johnsson, Lennart [$1.00]

3805:TR:80 "SSP Annual Report,"; Silicon Structures Project [$3.00]

3762:TR:80 "A Software Design System," Ph.D. Thesis; Hess, Gideon [$8.00]

3761:TR:80 "A Fault Tolerant Integrated Circuit Memory," Ph.D. Thesis; Barton, Tony [$7.00]

3760:TR:80 "The Tree Machine: A Highly Concurrent Computing Environment," Ph.D. Thesis,; Browning, Sally [$10.00]

3759:TR:80 "The Homogeneous Machine," Ph.D. Thesis; Locanthi, Bart [$7.00]

3710:TR:80 "Understanding Hierarchical Design," Ph.D. Thesis; Rowson, James [$9.00]

3642:TM:80 "Modeling and Verification in Structured Integrated Circuit Design," Ph.D. Thesis; Buchanan, Irene [$10.00]

3487:TM:80 "The Proposed Sticks Standard,"; Trimberger, Steve [$2.00]

3364:TM:79 "Stack Data Engine," December 1979; Efland, G. and R. C. Mosteller [$5.00]

3357:TM:79 "CIF20P Instruction Manual,"; Tarolli, Gary and Dick Lang [$3.00]

3356:TR:79 "LAP User's Manual,"; Lang, Dick [$3.00]

3353:TM:79 "FORTRAN Debugging Aids,"; Trimberger, Steve [$3.00]

3352:TM:80 "A Comprehensive CIF Test Set," December 1979; Trimberger, Steve [$2.00]

2883:TR:79 "A Pascal Machine Architecture Implanted in Bristle Blocks, a Prototype Silicon Compiler," MS Thesis; Seiler, Larry [$10.00]

2870:TM:79 "A Wire Oriented Mask Geometry Editor,"; Trimberger, Steve [$5.00]

2686:TR:80 "The Caltech Intermediate Form for LSI Layout Description,"; Sproull, Robert and Richard Lyon, revised by S. Trimberger [$2.00]

2276:TM:78 "A Language Processor and a Sample Language,"; Ayres, Ron [$12.00]

1584:TM:78 "Cost and Performance of the VLSI,"; Mead, Carver A. [$3.00]

1438:TM:78 "Polygon Package,"; Sutherland, Ivan E. [$3.00]